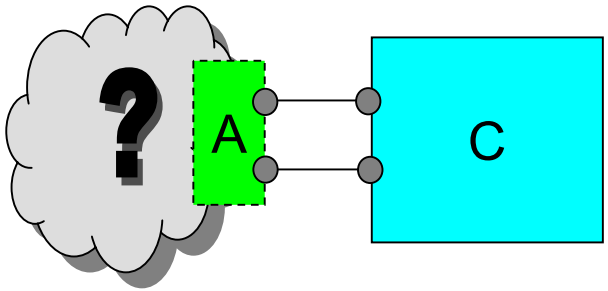
Component Assumptions

Problem: In what environments does component C satisfy safety property P?



- C is an open system, i.e., interacts with some environment
- The environment may be unpredictable or unknown
- Component C does not satisfy P in all environments
- Can helpful environments be characterized?

Solution: Developed novel approach to generate "assumptions" automatically

- Assumptions are abstract characterizations of all helpful environments in which C satisfies P
- Assumptions are *weakest*; they do not impose more restrictions than necessary to the environment
- Can be used as runtime monitors to trigger recovery actions when environment behaves differently
- Applied to Ames K9 Rover executive

Awards

- ASE 2002 best paper (September 2002), ACM distinguished paper award

What is new

- Algorithm extended to handle deadlock
- Applied to DS-1 remote agent executive; it detected the known bugs.
- Submitted (Jan 2003) extended version to Journal of Automated Software Engineering (by invitation)

Explanation

- **POC:** Dimitra Giannakopoulou and Corina Păsăreanu
- Joint work with Howard Barringer
- **Shown on slide:** Verification of large systems requires a divide and conquer approach; system properties are checked in terms of properties of components. However, the correctness of a component C typically depends on its environment, i.e., the behavior of components with which it interacts. Therefore, when verification of a component returns that a required property is not satisfied, it may be because the model of the environment used was not appropriate. In most cases, coming up with appropriate environment models is a difficult manual task. We have developed an algorithm that automatically generates abstract characterizations, called assumptions, of those environments in the context of which the component behaves as required. These assumptions are “weakest”, which means that they restrict the behavior of the component’s environment no more and no less than is necessary. For unpredictable environments, assumptions can be used as runtime monitors during deployment, and trigger special recovery actions when the environment does not behave as expected.
- **Accomplishment:** Our algorithm has been implemented in the LTSA tool for design-level model checking. It has been successfully applied to a number of case studies, including the Mars K9 Executive prototype. This work was presented at the ASE 2002 conference, and received an ACM distinguished paper award, as the best conference paper.
- **New accomplishments:** We have recently extended our approach for deadlock. The idea is, in addition to safety violations, to also avoid deadlocking the component in the context of the assumptions that our algorithm generates. We have applied the extended approach at the design-level of the DS-1 remote agent executive. We extended our ASE paper into a journal version submitted by invitation to the journal of Automated Software Engineering.
- **Future Plans:** Include evaluating our approach in the context of larger systems, improving the efficiency and memory usage of our algorithms, and investigating assumption generation in the context of actual software code rather than designs.